ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/273948933

Communities of Interacting Automata

Article *in* Doklady Mathematics · November 2003

CITATION
1

READS
84

1 author:

Irina A. Lomazova
National Research University Higher School of Economics
132 PUBLICATIONS **1,133** CITATIONS

SEE PROFILE

All content following this page was uploaded by Irina A. Lomazova on 24 March 2015.

The user has requested enhancement of the downloaded file.

—— COMPUTER SCIENCE ——

Communities of interacting automata

I. A. Lomazova

Finite automata [2] are widely used in modelling discrete systems, especially when handling control aspects is important. The benefits of finite automata are in particular connected with clear mathematical semantics, systems visualization and decidability of many crucial behavioral properties. The up-to-date needs for modelling concurrent and distributed systems with a complex structure lead to different extensions of the finite automata formalism, such as hierarchical, interacting, dynamic automata, which allow to model hierarchy, concurrency and communication among concurrent components [3, 6, 7].

Here we introduce the systems of *interacting automata*, which together with concurrency and communication allow to model dynamic activation of new agents. There is a rather large variety of formalisms for representing distributed systems in the modern theoretical computer science. Modelling systems with a dynamically changing number of agents requires extending standard formalisms to capture a possibility of changing a system structure (see e.g. [4] for an extension of Petri nets for modelling multi-agent systems).

Interacting automata represent in some sense the most simple model of a system consisting of agents, which can do autonomous actions, interact with each other, be generated by other agents and die. Interacting agents form an unstructured community with no hierarchical or any other structure, all agents have thus equal rights to do actions autonomously or synchronize. Agents are represented by nondeterministic finite automata. Initially there is one (parent) automata which can generate new members of the community, which in turn can also generate new members. An agent automata dies, when it comes to its terminal state. Concurrent behavior is represented by the interleaving semantics. A community

comes to its final state, when it is empty, i.e. all its members have died.

Now we come to the precise definitions.

Definition 1. A nondeterministic finite automaton is defined as a tuple $A = (Q, U, \theta, q_0, F)$, where
- Q is a finite set of inner states;
- U is a finite alphabet of action names;
- $\theta : (Q \setminus F) \times U \to 2^Q$ is a (partial) transition map;
- $q_0 \in Q$ — an initial state;
- F — a finite set of terminal states, $F \cap Q = \emptyset$

A finite automaton can be considered as a finite oriented graph with a set $Q \cup F$ of nodes and with arcs labelled by symbols from U. As usual, we suppose U contains a special symbol ϵ for the silent (invisible) action. In the graph representation inner states are designated by circles. The set Q of inner states contains a designated initial state and several terminal states.

A community of interacting automata will be a multiset of finite automata with actions of three types: usual autonomous actions, actions of synchronization, and actions generating finite automata — new members of the community.

Recall, that a *multiset* m over a set S is a mapping $m : S \to \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers, i.e. a multiset may contain several copies of the same element. A multiset m is *finite* iff the set $\{s \in S \mid m(s) > 0\}$ is finite. For two multisets m, m' the sum and the subtraction are defined as usual: $\forall s \in S : (m + m')(s) = m(s) + m'(s)$; $(m - m')(s) = m(s) - m'(s)$, if $m(s) > m'(s)$, and $(m - m')(s) = 0$, otherwise.

Definition 2. Let L be a finite set of synchronizing labels, s.t. for each label $l \in L$ the adjacent label $\bar{l} \in L$ is defined and $\bar{\bar{l}} = l$. Let U be a finite set of action names, $\mathcal{A} = \{\alpha_1, \ldots, \alpha_k\}$ — a set of automata

names. *A community of interacting automata* is defined as a collection $\Sigma = (A_1, \ldots, A_k)$ $(k \geq 1)$ of finite nondeterministic automata, where for $i = 1, \ldots, k$ an automaton $A_i = (\alpha_i, Q_i, (U \cup L \cup \mathcal{A}), \theta_i, q_0^i, F_i)$ is specified by the following components:

- α_i — a name of the automaton A_i;
- Q_i — a finite set of inner states of the automaton A_i;
- $(U \cup L \cup \mathcal{A})$ — a (common for all automata) set of names for autonomous (U), synchronized (L) and generating $(\mathcal{A})$ actions;
- $\theta_i : Q_i \times (U \cup L \cup \mathcal{A}) \to 2_i^Q$ — a transition map for A_i;
- $q_0^i \in Q_i$ — an initial state for A_i, and
- F_i — a set of terminal states for A_i.

Among this collection a distinguished *parent automaton* is chosen, we suppose it to be A_1.

Now we define the behavior of an automata community $\Sigma = (A_1, \ldots, A_k)$.

A *configuration of an automaton* A_i is represented by a pair (α_i, q), where α_i is the name, $q \in Q_i \cup F_i$ — a state of A_i. An automaton configuration (α_i, q) is called *terminal*, iff $q \in F_i$ — a terminal state. A *configuration of an automata community* Σ is a multiset K over the set of all nonterminal configurations of automata $A_1, \ldots, A_k$. A *step* for an automata community Σ is a triple (K, u, K'), where K, K' — configurations for Σ, $u \in (U \cup \mathcal{A} \cup \{\epsilon\})$ — an action name. We define three kinds of steps.

An autonomous step. Let the configuration K include an automaton configuration (α_i, q) and for some $u \in U$ we have $q' \in \theta_i(q, u)$, i.e. the autonomous action u transforms the state q of the automaton A_i into the state q'. Then the configuration K of Σ is transformed to K' (write $K \xrightarrow{u} K'$), where $K' = K - (\alpha_i, q) + (\alpha_i, q')$ provided $q' \in Q_i$ and $K' = K - (\alpha_i, q)$, if $q' \in F_i$. Thus terminal automaton configurations are deleted from the community.

A synchronization step. Let K include two automata configura-

tions (α_i, q), (α_j, p) and for some label $l \in L$, $q' \in \theta_i(q, l)$, $p' \in \theta_j(p, \bar{l})$. Then K can be transformed to K' (write $K \xrightarrow{\epsilon} K'$), where K' is obtained by deleting all terminal automaton configurations from $K - (\alpha_i, q) - (\alpha_j, p) + (\alpha_i, q') + (\alpha_j, p')$.

A generating step. Let K include an automaton configuration (α_i, q) and for some automaton name $\alpha_j \in \mathcal{A}$ we have $q' \in \theta_i(q, \alpha_j)$. Then K can be transformed to K' (write $K \xrightarrow{\alpha_j} K'$), where K' is obtained by deleting a terminal automaton configuration (if any) from $K - (\alpha_i, q) + (\alpha_i, q') + (\alpha_j, q_0^j)$.

We write $K \to K'$, if there exists a step from K to K'.

An automata community behavior is defined in terms of *consecutive runs* (or just *runs*). A run R is a finite or infinite sequence of automata community configurations $K_0, K_1, \ldots$, such that

1. The *initial configuration* K_0 consists of the only automaton configuration (α_1, q_0^1), i.e. it contains the parent automaton in its initial state;

2. $K_i \to K_{i+1}$ for all $i = 0, 1, \ldots$;

3. R is finite iff it can't be continued. Thus the last configuration in a finite run may be either empty (a terminating run), or a nonempty configuration in which no action can occur (a deadlock).

The *behavior* of an automata community Σ is defined as the set of all its runs.

Let Σ be an automata community. The language $\mathcal{L}(\Sigma)$ accepted by Σ is defined as the set of all finite words over the alphabet $(U \cup \mathcal{A}) \setminus \{\epsilon\}$, s.t. for each word $w \in \mathcal{L}(\Sigma)$ there exists a terminating run R for Σ with a sequence w of visible actions.

It is well-known, that the class of languages accepted by nondeterministic finite automata coincides with the class of regular languages and strictly included into the class of context-free languages, accepted by push-down automata. In particular, the language $\mathcal{L}_1 = \{w \mid w \in \{a, b\}^*, n_a(w) = n_b(w)\}$, where $n_a(w)$ and $n_b(w)$ designate the number of symbols 'a'

(symbols 'b' correspondingly) in the word w, forms an example of a context-free language, which can't be accepted by any finite nondeterministic automaton. The language $\mathcal{L}_2 = \{a^n b^n c^n \mid n \geq 1\}$ is in turn an example of a noncontext-free language, which thus cannot be accepted by any push-down automaton.

Proposition 1. *There exist two communities of interacting automata* Σ_1 *and* Σ_2*, accepting the languages* $\mathcal{L}_1 = \{w \mid w \in \{a, b\}^*, n_a(w) = n_b(w)\}$ *and* $\mathcal{L}_2 = \{a^n b^n c^n \mid n \geq 1\}$ *correspondingly.*

The following theorem is a consequence of this proposition.

Theorem 1. *The communities of interacting automata are strictly more expressive than the nondeterministic finite automata, and are not less restrictive than the push-down automata.*

Now we compare automata communities with Petri nets — a popular formalism for modelling and analysis of parallel and distributed systems [1].

Definition 3. A *Petri net* is a finite directed graph with two types of nodes, referred to as *places* and *transitions*. An arc in a Petri net goes either from a place to transition, or from a transition to a place. For a transition t every place p with an arc going from p to t is called an *input place*, and a place q with an arc going from t to q is called an *output place*. A *marking* M in a Petri net is a mapping from the set P of places to nonnegative integers. We say, that there are k tokens in the place p, if $M(p) = k$. A *marked Petri net* is a Petri net together with its initial marking.

We now define the behavior of a Petri net. A transition t is *enabled* in a marking M iff each input place of t contains at M at least one token. An enabled transition t may *fire* by removing one token from every input (for t) place and by adding one token to every its output place. A Petri

net *run* is defined as a sequence of its firings starting from the initial marking.

Transitions in a Petri net may be labelled by names of actions to make a behavior of a Petri net observable. Some of transitions may be invisible (labelled by a silent action ϵ).

The observable behavior of a Petri net can be described in terms of Petri net languages.

Definition 4. Let *PN* be a labelled Petri net with transition labels from X. Let $\mathcal{R}_\emptyset$ be the set of all runs for *PN*, leading to the empty marking. A *terminal language* of *PN* is a set of all words (over X) labelling runs from $\mathcal{R}_\emptyset$ (invisible actions are not taken here into account).

Now we compare expressibility of Petri nets and interacting automata communities. The following theorem states that for every community of interacting automata Σ there exists a Petri net $PN(\Sigma)$ simulating its behavior.

Theorem 2. *For every automata community* Σ *there exists a Petri net* $PN(\Sigma)$*, such that the terminal language of* $PN(\Sigma)$ *coincides with the language* $\mathcal{L}(\Sigma)$*, accepted by the automata community* Σ*.*

It can be shown, that for each run of Σ there exists a corresponding run of $PN(\Sigma)$ with the same sequence of visible actions and vice versa. The proof can be done by induction on the length of the run by the analysis of all cases of state transitions.

In the theory of Petri net languages it is known, that the context-free languages are incomparable with the Petri net languages [1]. More precisely, there exist some context-free languages, which are not terminal Petri net languages. An example of such a language is the language $\{ww^R \mid w \in \{a, b\}^*\}$, where w^R denotes a word w in the inverse order, i.e. written from the right to the left. Then from the theorem 2 we get the following

Theorem 3. *There exists a context-free language, which is not accepted by any automata community.*

From this theorem and the theorem 1 it follows that interacting automata communities and push-down automata are incomparable by expressibility, i.e. these two classes of languages intersect, but neither of them is included into the other.

Recall, that the *Reachability Problem* is to decide, given an initial state S and a finite state S', whether there exists a run, leading from S to S'. A Petri net $PN(\Sigma)$ simulates all runs of Σ not only as sequences of actions, but also as sequences of reachable states. These means, that the Reachability Problem for automata communities can be reduced to the Reachability Problem for Petri nets. For Petri nets the reachability is decidable [5], hence we obtain

Theorem 4. *Reachability Problem is decidable for interacting automata communities.*

This research was supported by the Presidium of the Russian Academy of Science, program "Intellectual computer systems" (project 2.3 – "Instrumental software for dynamic intellectual systems") and the Russian Fund for Fundamental Research (grant 03-01-00804).

View publication stats

References

[1] Kotov V.E., *Seti Petri* (Petri nets), Moscow: Nauka, 1984.

[2] Hopcroft, J. E., Motwani, R., and Ullman, J. D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publ. Co., 2001.

[3] Alur, R., Kannan, S. and Yannakakis, M., *Lecture Notes in Computer Science*, 1999, vol. 1644, pp. 169–178.

[4] Lomazova, I. A., *Fundamenta Informaticae*, 2000, vol. 43, no. 1–4, pp. 195–214.

[5] Mayr, E. W., *SIAM Journal on Computing*, 1984, vol. 13, pp. 441-460.

[6] Trakhtenbrot, B. A., *Lecture Notes in Computer Science*, 2001, vol. 2076, pp. 4–23.

[7] Yannakakis, M., *Lecture Notes in Computer Science*, 2000, vol. 1875, pp. 315–330.